This plan is designed to improve programing fundamentals, especially in Python. The goal is to show master of the objectives below. Please commit all code to a repo as it is completed.

**The Plan:**

1. Read [Computational Thinking](https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf)
2. [Codecademy’s Python Course](https://www.codecademy.com/learn/python)
3. [Introducing Python](http://shop.oreilly.com/product/0636920028659.do) Ch. 1-3
4. [Codecademy’s Command Line](https://www.codecademy.com/en/courses/learn-the-command-line)
5. [Ryan’s CLI tutorial](http://ryanstutorials.net/linuxtutorial/)
6. [Code School for git](https://try.github.io/levels/1/challenges/1)
7. [Udacity’s Git course](https://www.udacity.com/course/how-to-use-git-and-github--ud775)
8. [Introducing Python](http://shop.oreilly.com/product/0636920028659.do) Ch. 4-8
9. [MIT OCW Unit 1](http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/) Readings 2&3, Lecture - Chapter 1 OOP and Design Lab 1
10. [Introducing Python](http://shop.oreilly.com/product/0636920028659.do) Ch 9-10
11. [Data Science from Scratch](http://shop.oreilly.com/product/0636920033400.do) Ch 1-6

**Programming Objectives**

|  |
| --- |
| Understand and apply fundamental programming concepts |
| Explain why we are using Python for Data Science. List 4 advantages and 4 disadvantages of Python for Data Science. |
| Sequences - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Iterators - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Functions - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Classes - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Context managers - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Special method names - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Syntax - Give a plain English definition, write correct Python syntax, and write wrong Python syntax. |
| Strongly Typed - Give a plain English definition. Explain how Python is strongly typed. |
| Polymorphism - Give a plain English definition. Write an example of polymorphic Python code. |
| Specification - Give a plain English definition. Write an example of a specification. |
| Test Driven Development (TDD) - Give a plain English definition. Write an example of a TDD in Python. |
| Red, Green, Refactor - Give a plain English definition, give general use cases, and write Python code that applies the concept |
| Comments - Give a plain English definition, give general use cases, and write Python code that applies the concept |
| docstrings - Give a plain English definition, give general use cases, and write Python code that applies the concept |
| docstring test - Give a plain English definition, give general use cases, and write Python code that applies the concept. |
| Use Jupyter Notebook as a development environment. Write Python code, make a figure, write markdown, and write MathJax. |
| Be able to display docstring in Jupyter Notebook. |
| Given a mathematical formula convert it to Python code. |
| Given pseudo code convert it to Python code. |
| Given a programming specification write the Python code, including test. |
| Given a logical statement convert it to Python code. |
| Explain best practices around pair programming. |
| Explain the pros and cons of style guidelines. |
| Name, give examples, and compare at least 3 different runtime orders of magnitude (e.g., constant, factorial) |
| Write code to profile and benchmark code. |
| Refactor code to improve runtime performance (e.g., perform set lookup instead of list lookups). |
| Explain what idiomatic Python is. Refactor code to be idiomatic. |
| Explain the major differences between Python 2 and Python 3. Write an example between the two. |
| Explain what NumPy is. Write code that uses NumPy data structures and functions. |
| Explain what Pandas is. Write code that uses Pandas data structures and functions. |
| Install and use a 3rd party package from a package manager. |
| Define OOP. Provide examples. Provide pros and cons. |
| Define functional programming. Provide examples. Provide pros and cons. |
| Define metaprogramming. Provide examples. Provide pros and cons. |
| Be able to debug with print statements. |
| Be able to debug with pdb. |
| Write tuple unpacking. |
| Write multiple assignment. |
| Run script at the command line with arguments. |